Федеральное государственное автономное
Образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт Космических и информационных технологий
Базовая кафедра геоинформационных систем

УТВЕРЖДАЮ
Заведующий кафедрой

_______ В.И. Харук
«____» _____2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 – Информационные системы и технологии

Картографирование дендроклиматических показателей на основе микропараметров клеток

Руководитель _____ доцент каф. Б-ГИС, к.т.н. А.С. Савельев

Выпускник _____ Г.А. Мамышева

Нормоконтроллер _____ Е.В. Федотова

Красноярск 2018

Федеральное государственное автономное
Образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт Космических и информационных технологий
Базовая кафедра геоинформационных систем

УТВЕРЖДАЮ
Заведующий кафедрой

_____ В.И. Харук
«____» _____2018 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Мамышевой Галине Алексеевне

Группа КИ14-14Б, направление 09.03.02 Информационные системы и технологии.

Тема выпускной квалификационной работы «Картографирование дендроклиматических показателей на основе микропараметров клеток».

Утверждена приказом по университету № 7461/с от 24.05.2018 г

Руководитель ВКР А. С. Савельев, доцент кафедры Б-ГИС.

Исходные данные для ВКР: данные о структурных параметрах клеток хвойных деревьев в формате excel файлов

Перечень разделов ВКР: введение; анализ предметной области, инструменты и алгоритмы для реализации веб-приложения, реализация руководство пользователя, заключение, список использованных сокращений, список использованных источников, приложение.

Перечень графического материала: слайды презентации.

Руководитель ВКР _____ А. С. Савельев

Задание принял к исполнению _____ Г. А. Мамышева

«____» _____ 2018

РЕФЕРАТ

Выпускная квалификационная работа по теме «Картографирование дендроклиматических показателей на основе микропараметров клеток» содержи 58 страниц текстового документа, 1 приложение, 14 использованных источников.

ДЕНДРОКЛИМАТИЧЕСКИЙ АНАЛИЗ, ДЕНДРОХРОНОЛОГИЯ, ВЕБ-КАРТОГРАФИРОВАНИЕ, ТРАХЕИДЫ, ЛЮМЕН.

Цель работы: создание веб-приложения способного визуализировать данные о микропараметрах клеток деревьев с возможностью вывода данных по отдельным деревьям и агрегирования данных по выбранным параметрам для сопоставления с климатическими условиями в заданных участках.

Задачи работы:

А) сформировать реляционную базу данных с данными о микропараметрах клеток и областях расположения срезов деревьев;

Б) написать серверную часть для получения запросов от клиента, получения данных из базы данных и отправки данных клиенту;

В) написать клиентскую часть, визуализирующую на карте данные о микропараметрах клеток из базы данных;

Г) реализовать функционал, позволяющий агрегировать данные по областям и визуализировать по отдельным параметрам клеток.

В работе картографировались данные о временной изменчивости параметров клеточной структуры.

В результате выполнения работы было реализовано веб-приложения, визуализирующее по координатам места срезов деревьев и выводящее их информацию на своей странице.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современное состояние биосферы является результатом нелинейного взаимоотношения естественных и антропогенных причин. В основе исследований лежит представление о дендрохронологии и дендроклиматическом анализе, как одни из наиболее приоритетных проблем в науке об экологии и имеют большое практическое значение.

Исследования в области дендрохронологии и дендроклиматологии обрели мировую известность благодаря работам двух научных школ – Е. А. Ваганова и С. Г. Шиятова, сформировавшихся в Институте леса им. В. Н. Сукачева СО РАН, а также Институту экологии растений и животных УрО РАН.

В начале 90-х годов в России была создана государственная система экологического мониторинга, в состав которой входит раздел дендроклиматического и дендрохронологического мониторинга, то есть «информационная система слежения, оценки и прогноза изменений годичного прироста деревьев и определяющих этот прирост факторов» [1].

На сегодняшний день было собрано большое количество данных различных пространственно-временные и распределительные дендроклиматические данные. Они отображают состояние лесных экосистем под влиянием климатических факторов природы.

Основная научная проблема, частью которой является предлагаемый проект – оптимизация анализа областей среза деревьев.

Одно из возможных применений разрабатываемого веб-приложения – использование в гранте Шишова В. В. 14-14-00219 «Экспериментально-теоретический анализ изменчивости роста древесных растений в континентальной части Сибири (Енисейско-Ленский трансект)».

Целью настоящей работы является создание веб-приложения способного визуализировать данные о микропараметрах клеток деревьев с возможностью вывода данных по отдельным деревьям и агрегирования

данных по выбранным параметрам для сопоставления с климатическими условиями в заданных участках.

Для достижения цели необходимо решить следующие задачи:

- сформировать реляционную базу данных с данными о микропараметрах клеток и областях расположения срезов деревьев;

- написать серверную часть для получения запросов от клиента, получения данных из базы данных и отправки данных клиенту;

- написать клиентскую часть, визуализирующую на карте данные о микропараметрах клеток из базы данных;

- реализовать функционал, позволяющий агрегировать данные по областям и визуализировать по отдельным параметрам клеток.

1 Анализ предметной области

1.1 Анализ методов картографирования

С развитием новых технологий появилось огромное количество средств и инструментов, которые значительно облегчают создание веб-картографических приложений.

Одним из самых простых способов создания и публикаций пространственных данных это использование *виртуальных глобусов*. Виртуальные глобусы – программное обеспечение с помощью которого можно работать с трехмерной моделью земли [2]. Эта группа инструментов позволяет визуализировать данные с привязкой к географическим координатам. Отличаются тем, что распространение является глобальным и стремительной доставкой данных пользователю. Одно из преимуществ виртуальных глобусов заключается в том, что они включают доступ к некой «подложке» - базе данных [3]. Также это может быть существенным недостатком, так как зачастую данную подложку сменить невозможно. Кроме того, виртуальным глобусам характерны трудности работы с большими объемами пользовательских данных, а также возможны проблемы с настраиванием. К виртуальным глобусам можно отнести такие приложения как Google Maps, Google Earth, Virtual Earth, ArcGIS и Sputnik GIS.

Пользовательские ГИС предоставляют клиенту интерфейс для работы с геоданными, а также позволяет осуществлять подготовку и анализ данных перед публикацией в веб. К пользовательским ГИС можно отнести такие системы как ArcGIS, Mapinfo, QGIS, gvSIG.

Картографические сервисы создает карты, объекты и данные атрибутов внутри многих типов клиентских приложений [4]. Это способ, предназначенный для быстрой публикации данных в Интернете, а также обработки необходимых запросов. Серверные ГИС предоставляют возможность создать пользовательский интерфейс, а также связать сервис с

базой данных. Базы данных должны поддерживать классы пространственных данных. Для работы с веб-сервисами необходимы навыки программирования на языке JavaScript или PHP.

Помимо простой визуализации и создания данных, самым новым аспектом работы с пространственными данными является перенос в веб их обработки и анализа. Это становится возможным благодаря мощному развитию инструментария легко размещаемого на веб-серверах, как открытого GDAL, PROJ, GeoTools, FDO, так и проприетарного ArcGIS Server [5].

Картографическое моделирование относится к процессу, в котором производятся несколько тематических слоев той же области, обрабатывается и анализируется. Операции над слоями карты можно объединить в алгоритмы, и в конечном итоге в моделирование или модели оптимизации [6].

1.2 Обзор исходных данных

Изменения условий среды в природе за длительные периоды времени можно изучать только по косвенным источникам. Годичные кольца деревьев обладают несколькими значительными преимуществами [7]. Во-первых, на данный момент существенно развиты теоретические и методические основы дендрохронологии [8]. Во-вторых, годичные кольца являются не только интеграторами на внешние условия, но еще и регистрируют изменения скорости роста под воздействием этих условий. В-третьих, формирование клеток годичного кольца происходит в течении всего сезона, что позволяет исследовать изменения среды с гораздо большим временным разрешением, чем год.

Наиболее легко измеряемым параметром является ширина годичного кольца. Данный параметр пропорционально числу клеток в годичном кольце и обладает высокой чувствительностью к внешним условиям от ранней весны и химических загрязнений до извержения вулканов, и падения метеоритов.

Трахеиды (прозенхимные клетки) – сильно вытянутые клетки с утолщенными стенками, составляющие более 90% объема древесины. Ряды трахеид внутри годичных колец представляют регулярные изменения размеров клеток и толщины их клеточных стенок. Их изменчивость отражает влияние внутренних и внешних факторов на рост древесных растений. Важно получать и анализировать информацию об изменении размеров клеток в годичных кольцах. Проще всего такие измерения можно осуществить на тонких поперечных срезах древесины с помощью микроскопа с точностью линейной подачи образца в 0.5 мкм или лучше [9]. Связь клеточных размеров с климатическими параметрами среды обитания изложены в работах Ваганова и др. [10].

Основными измеряемыми структурными элементами трахеид являются радиальный и тангенциальный размеры люмена и клетки, толщина одинарной и двойной стенки (рис. 1).

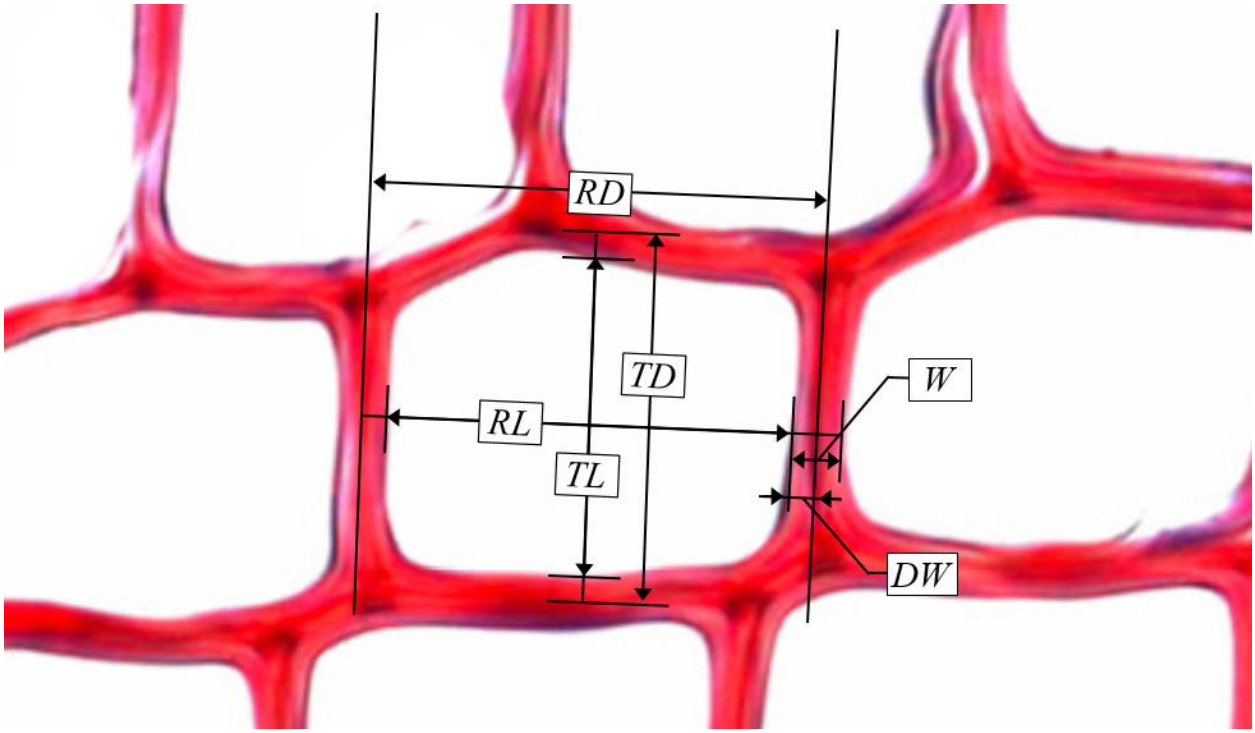

Рисунок 1 – Измерения трахеид хвойных деревьев. RL, TL – радиальный и тангенциальный размеры люмена, RD, TD – радиальный и тангенциальный размер клетки, W, DW – размер одинарной и двойной стенки

Для реализации программы были использованы 4 структурных параметра, изображенных в таблице 1

Таблица 1 – Список параметров в системе

Название параметра	Описание
Ширина люмена	Радиальный размер люмена клетки
Высота люмена	Тангенциальный размер люмена клетки
Площадь люмена	Площадь области люмена клетки
Двойная стенка	Длина области между двумя соседними клетками

Так как структура годичных колец характеризуется рядом показателей, то влияние внешних факторов необходимо рассматривать каждый показатель в отдельности и их совокупность. К настоящему моменту опубликовано несколько монографий, где подробно описано влияние внешних факторов на изменчивость ширины годичных колец.

1.3 Обзор картографических веб-сервисов

Среди общего числа картографических сервисов наиболее важные для данного являются сервисы, работающие по протоколу WMTS. Именно в этом стандарте работают веб-сервисы Яндекс Карты, OpenStreetMap и Google Maps. Все эти сервисы предоставляют API для работы с ними через фрейм, на который будет получаться карта.

Яндекс Карты – качественный сервис, предоставляющий снимки со спутника с высоким разрешением. Имеет наилучшие результаты в качестве снимков по России, но уступает Google Maps по картам мира. Имеет лучшую детализацию в городах с населением меньше 1 млн людей. Имеет несколько видов карт: карта, снимок, гибрид.

Яндекс Карты предоставляют API для работы с сервисом при помощи JavaScript. Данное API позволяет выполнять основные действия с картой, такие как: добавление маркера на карту, кластеризация маркеров, позиционирование, масштабирование и др. Все элементы карты располагаются на отдельных векторных слоях, что позволяет накладывать на карту различные фильтры и дополнительные карты.

OpenStreetMap – некоммерческий веб-сервис, направленный на создание схематических карты. Данный веб-сервис является одним из самых старых веб сервисов. Изначально он разрабатывался для построения схем дорог и их предоставление на карте мира. Данный сервис появился еще до появления Яндекс Карт и Google Maps. Данный проект использует принцип вики, то есть каждый пользователь может зайти и добавить какую-либо информацию и существует за счет пожертвований.

OpenStreetMap не имеет определенного API для работы с ним, но наиболее распространенным считается OpenLayers. Он дает возможность размещать динамические карты на любом сайте, позволяет выводить плитки OSM или векторные слои, а также создавать и расширять элементы карты.

Google Maps является наиболее распространенным картографическим сервисом, предоставляющего интерактивные спутниковые снимки в режиме онлайн. Данный сервис предоставляет высококачественные снимки с космоса с высоким разрешением, что позволяет делать интерпретацию карты наиболее качественно. Кроме спутниковых карт Google Maps имеет схематическую карту. Все данные с данного сервиса являются бесплатными и открыты в общем доступе.

Google Maps API (JavaScript API) предоставляют широкий спектр возможностей. API позволяет строить маркеры, кластеризации, визуализации карт, позиционирования, информационные окна и мн. Другое. При этом система построена так, что для выполнения того или иного действия требуется минимум кода, а если будут необходимы дополнительные возможности или действия, то они задаются в опциях, при создании объектов.

Данный сервис является всемирно известным, имеет удобный и качественный API и является наиболее оптимальным для разрабатываемого приложения.

1.4 Обзор языков для реализации серверной части

Для реализации серверной стороны имеется множество различных языков и Фреймворков, добавляющих те или иные возможности. Будут рассмотрены наиболее часто используемые языки и Фреймворки.

ASP.NET на языке C# является единой моделью для разработки веб-приложений. Одно из главных достоинств данного Фреймворка является объектно-ориентированная парадигма программирования. Удобства предоставляют так же существующие Фреймворки для связи с SQL базами данных, например, Фреймворк Entity. Доступ к базе данных осуществляется за счет описанных классов и контекста, а получать данные можно при помощи LINQ функций, или SQL запросов. Минусы данной платформы является необходимость сериализации. Передача между разделами осуществляется только бинарными потоками, что может привести к замедлению работы сервера. Так же довольно сложно построить удобную систему, требуется писать много кода для описания каждого действия. При описании сервера на данном языке будут сложности в написании кода, поддерживании и редактировании при больших изменениях в проекте, но также данный язык предоставляет различные полезные механизмы для работы с базой данных, что могло бы упростить разработку приложения.

Python предоставляет возможности скриптового языка и хорошо совместим с JavaScript. Данный язык имеет различные Фреймворки для работы с вебом, например, Django или Flack. Данный язык имеет схожую с JavaScript структуру данных, что позволяет проще передавать данные между клиентом и сервером. Данный язык имеет библиотеку NumPy, скомпилированную на языке C++ и позволяющую выполнять быстрые

математические вычисления. Связь с базой данных выполняется через существующие модули для каждой из баз данных, а доступ к базе данных осуществляется через SQL запросы. Даже при наличии библиотеки NumPy – python все еще является скриптовым языком, что ведет к нагрузке в вычислениях.

JavaScript является языком, на котором описана функциональная часть клиентской части. Чаще всего в виде сервера на языке JavaScript используется Фреймворк NODE.js. Это тяжеловесный Фреймворк, позволяющий работать с JavaScript как с модульным языком, упрощающий работу связи между клиентом и сервером, например, управление сокетами. Главным достоинством написания серверной части на NODE.js – это то, что клиентская часть написана на том же языке, что означает отсутствие необходимости преобразовывать типы данных и передавать из в том виде, в каком они есть. Один из главных минусов является сложное управление памятью. Отсутствие нормальной работы сборщика мусора и постоянно растущие области видимости зачастую приводят к сбоям работы сервера. Кроме того, данный Фреймворк используется в языке R в виде реализации Фреймворка Shine, имеющего многие инструменты для визуализации и анализа данных.

Go – довольно молодой язык программирования, написанный компанией Google. Данный язык уже имеет множество пакетов для работы в большей части областей. Так, например, имеет пакеты для работы с базой данных, сериализацией и работой по протоколу HTTP. Данный язык имеет несколько достоинств. Главное его достоинство – скорость вычислений. Оно имеет значительно большую скорость, по сравнению с другими языками, описывающими серверную часть. Так же данный язык является компилируемым, что позволяет запускать сервер с одного exe файла. Третье достоинство – работа с потоками. В данной области очень сильно упростили эту работу. Процессы в данном случае не разбиваются на несколько потоков, а наоборот каждой задаче выделяется отдельный поток, а задачи

обрабатываются асинхронно. Для данного проекта это является одной из главных необходимостей.

1.5 Выводы по главе 1

В данной главе был проведен обзор методов картографирования, было выполнено описание и анализ Яндекс Карт, OpenStreetMap и Google Maps, а также выполнен обзор имеющихся языков программирования и Фреймворков для написания серверной части.

Выполняя анализ сведений было принято использовать на клиенте веб страницу, связанную с API Google Maps, так как нам наиболее необходим лесной покров карты мира и язык Go на серверной части, так как он наиболее подходит для данной задачи.

2 Инструменты и алгоритмы для реализации веб-приложения

2.1 Используемое программное обеспечение

2.1.1 Интегрированная среда разработки IDE JetBrains GoLang 2018

JetBrains GoLang – это интегрированная среда разработки для языка программирования Go. Данная IDE распространяется бесплатно для некоммерческого использования студентами. Пример интерфейса программы приведен на рисунке 2.

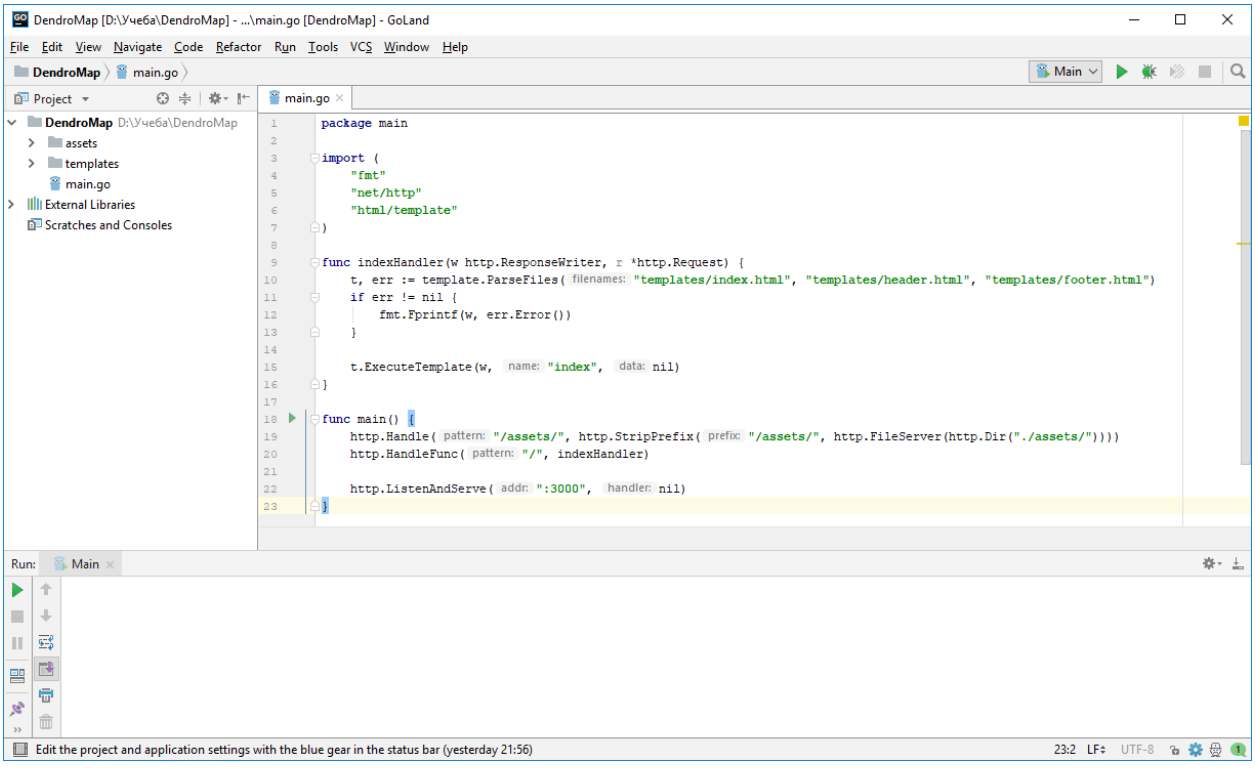

Рисунок 2 – Интерфейс IDE JetBrains GoLang

Данная IDE имеет проверку и подсветку синтаксиса, автозаполнение, построение кода компилятором и его выполнение. Так же данная среда имеет консоль вывода, возможность для отладки программы и возможность загружать дополнительные пакеты.

Помимо проверки синтаксиса для языка Go IDE поддерживает и Javascript, CSS, HTML, что будет использоваться для написания клиентской части приложения.

2.1.2 Пользовательский интерфейс pgAdmin 4

Для хранения данных приложения будет использоваться реляционная база данных, а для работы с ней – pgAdmin 4. PostgreSQL – это удобная СУБД для работы с веб приложениями, предоставляющая большой функционал и набор инструментов для работы с реляционными базами данных, что нам и необходимо. Пример интерфейса pgAdmin приведен на рисунке 3.

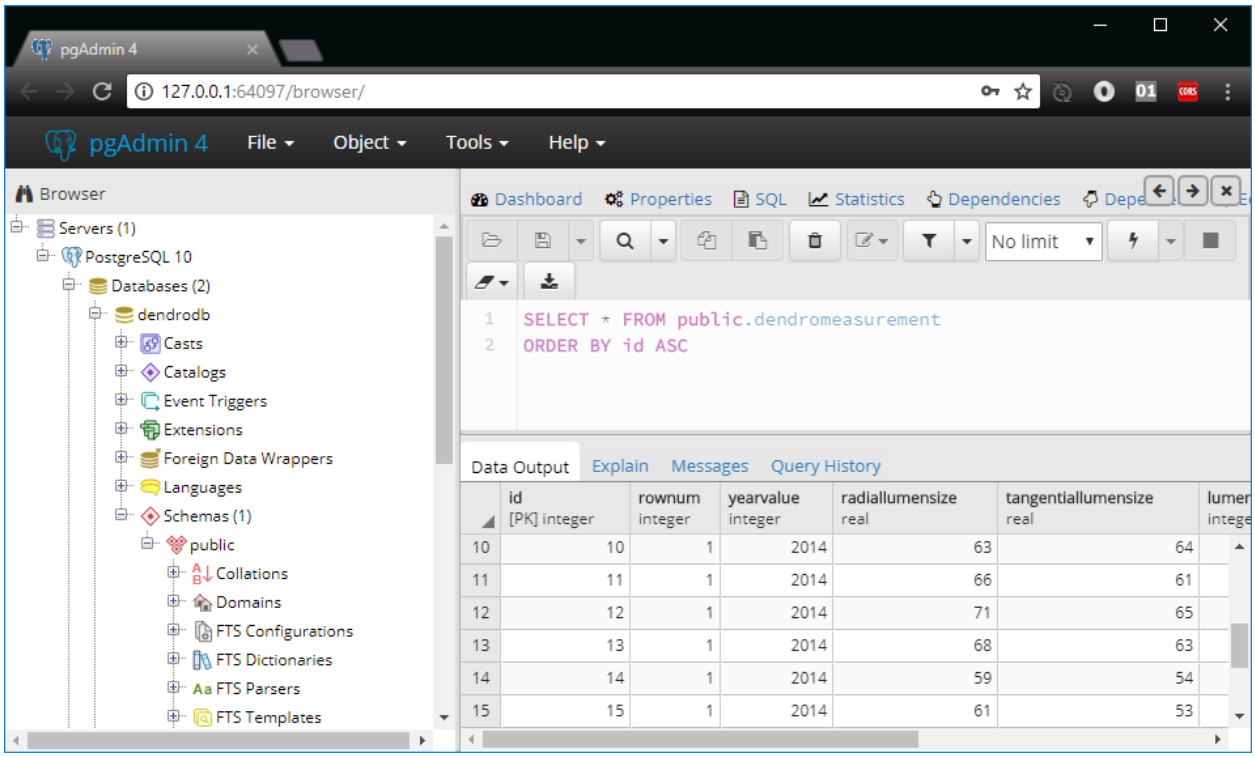

Рисунок 3 – Интерфейс pgAdmin 4

Данный интерфейс позволяет создавать базы данных, таблицы, роли, схемы, добавлять, изменять и удалять данные, использовать SQL запросы.

2.2 Используемые инструменты, библиотеки и языки программирования

2.2.1 Язык программирования Go

Для серверной части был выбран язык программирования Go по нескольким причинам. Первая причина – это скорость математических вычислений, например, скорость вычисления достигает отметки в 15 раз быстрее по сравнению со стандартными вычислениями скриптовых языков и более чем в 2 раза, по сравнению с php.

Так же данный язык позволяет легко работать с многопоточностью, при этом для каждой задачи формируются отдельные процессы, что не требует разбивания процессы на несколько частей для разделения вычислений на несколько процессоров, а для обобщения процессов используются каналы. Это значительно облегчает обработку запросов от множества пользователей одновременно.

Еще одним плюсом является работа с базой данных через SQL запросы и легкое преобразование ответов в JSON формат, что предоставляет наибольшую гибкость для работы с таблицами.

Чтобы запустить программу на сервере достаточно скопировать скомпилированный exe файл и зависимые файлы на сервер и запустить, без необходимости каких-либо дополнительных настроек, что так же является плюсом использования данного языка программирования для серверной части.

Сохраненная информация либо пространственные данные, либо соответствующие табличные данные могут быть извлекается с помощью языка структурированных запросов (SQL). В зависимости от типа пользовательский интерфейс, данные могут быть запрошены с использованием SQL или управляемой меню системы для извлечения картографических данных.

2.2.2 Языки HTML, CSS и JavaScript

JavaScript – это основной на данный момент язык программирования для добавления функционала веб страницам. Изначально он задумывался как язык, способный «оживить» страницу, но со временем это перешло на стороны CSS, а JavaScript стал полноценным языком, позволяющего делать почти все то же, что и остальные языки. Основную популярность язык получил в браузерах для придания интерктивности страницам.

Данные язык не поддерживает низкоуровневые операции, но имеет очень много возможностей на высоком уровне (создавать, изменять, удалять теги, синхронно или асинхронно загружать данные с других страниц, анализировать полученную информацию и др.). У языка JavaScript есть несколько очень важных плюсов: полная совместимость с HTML и CSS, поддерживаемость всеми основными браузерами и упрощение работы со страницами за счет DOM. Поэтому JavaScript – это наиболее распространенный язык для работы с интерфейсом браузера.

Каскадные таблицы стилей (Cascading Style Sheets = CSS) — это язык, отвечающей за визуальное отображение элементов страницы. Современный CSS способен работать с медиа-запросами и даже создавать анимации. При помощи CSS достигается адаптивная верстка, позволяющая строить страницы, работающими как на компьютере, так и на мобильном устройстве. Данная таблица стилей поддерживается всеми основными браузерами.

CSS – это язык стилей. Он не позволяет создавать классы, не обладает парадигмами, но благодаря этому идеально подходит для описания визуальных составляющих html элементов страницы. Использование CSS значительно облегчает создание сайтов, делает их более отзывчивыми и выдержанными в определенном стиле.

HTML (язык разметки гипертекста) – язык разметки веб страниц. Изначально он был задуман только для отображение текстовых данных на статичной странице, но со временем на нем стали написаны веб порталы,

приложения, видео и аудио проигрыватели, и др. Со временем стилестические составляющие перешли полностью в CSS, что позволили HTML создавать качественную структуру сайта. В текущее время поисковые системы распознают типы сайтов на основе структуры HTML страниц.

Любая веб страница состоит из элементов языка HTML. Каждый элемент сопровождается открывающим и закрывающим тегом, а вся информация сохраняется в первом из них. Со временем некоторые из тега стало возможно использовать в сокращенном виде, используя только один тег, а некоторые, такие как
 - совсем используются без закрывающего тега.

Из языка для разметки текста HTML стал языком для формирования всевозможных веб страниц, при том оставаясь все таким же простым, даже при наличии все растущего у него функционала.

2.2.3 Используемые инструменты и библиотеки

На языке Go используется несколько встроенных библиотек: «fmt», «strconv», «net/http», «html/template» и «encoding/json». Первая используется для форматируемого вывода в консоль, т.е. выводит информацию из любого объекта в читабельном виде. Вторая используется для конвертирования строки в число (при получении POST запроса с клиента). Библиотека «net/http» запускает сервер (слушателя) на заданном порту, получает запросы, отправляет ответы и выполняет форматирование запросов в соответствии с REST запросами. «html/template» собирает страницу из заданных шаблонов и возвращает результат на клиента. Последняя библиотека конвертирует полученные из базы данных в JSON формат и получает из полученного в запросе JSON файла данные в формате языка Go.

Из сторонних библиотек используются: «github.com/lib/pq» и «github.com/jmoiron/sqlx». Первая используется только для простоения связи Go с PostgreSql, поэтому инициализируется только ее конструктор. Вторая библиотека используется непосредственно для работы с базой данных:

подключение к БД, проверка соединения, выполнения запросов и т.д. В отличии от встроенной библиотеки sql данная позволяет формировать данные, полученные по запросу в PostgreSql в структуру данных, которая в последствии будет преобразована в JSON формат.

На стороне клиента используется API Google Maps v3 для получения слоя карты и добавления на него маркеров срезов деревьев. Google Maps представляют удобную API, без необходимости задавания всей информации вручную. Это работает за счет добавляемых модулей (например, стилей, всплывающих окон и т.п.), которые подключаются по мере их необходимости. Один из таких – модуль кластеризации, позволяющий при уменьшении маштаба объединять несколько маркеров в 1.

2.3 Кластеризация

Кластеризация представляет собой алгоритм объединения нескольких маркеров в один. При этом учитывается масштаб, при котором выполняется кластеризация и области, по которым проводятся кластеризации. Пример такой области является окружность, по которой выполняется объединение (рис. 4).

В данном случае расположение маркера будет выбрано по центру окружности. Данный способ качественной и точно выводит информацию о кластерах в заданных областях. Но при большом числе маркеров могут быть замечены существенные замедления системы, а также существуют сложности с выбором размера окружностей и сложности в вычислениях положений маркеров внутри нее.

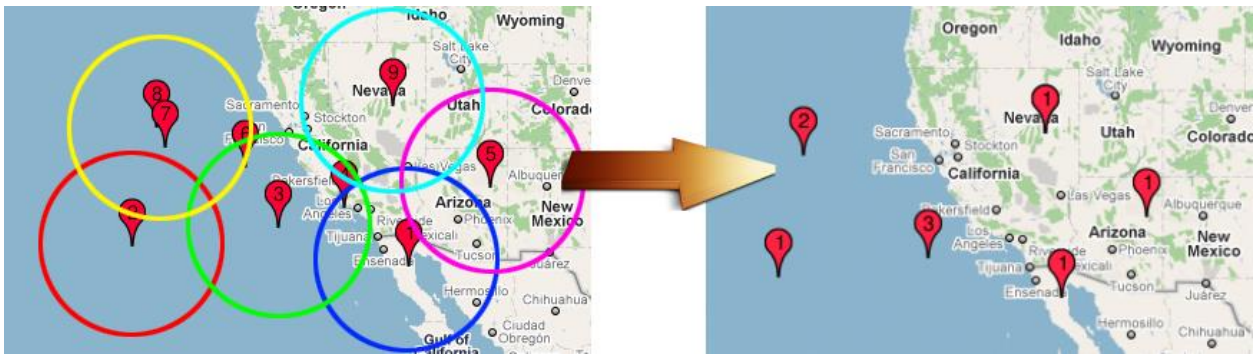

Рисунок 4 - Кластеризация по округлой области

Другой способ кластеризации – это использовать квадраты сетки карты, которая разбивает отображаемую карту на квадраты фиксированного размера (рис. 5).

Рисунок 5 – Кластеризация по квадратам

Это удобно, так как размер квадратов определен для каждого из масштабов предоставления карты. Такой способ существенно упрощает определение положения кластера, за счет расположения его по центру квадрата, определения маркеров, входящих в него, используя область, ограниченную четырьмя сторонами.

Данный способ работает значительно быстрее и качественно отбирает маркеры для кластеризации, что и необходимо для разрабатываемого веб-приложения. Единственный минус такого отображения – неточное позиционирование кластера относительно маркеров, чем можно пренебречь.

3 Реализация. Руководство пользователя

Разработанное веб-приложение должно предоставлять пользователю возможность отображения местоположения отдельных объектов (деревьев), а также иметь возможность проведения анализа и моделирования геоинформационных ресурсов.

Данная программа оптимизирует обработку запросов и формирования картографических данных, повышая качество проведения исследования.

3.1 Концепция веб-приложения

На основе На рисунке 6 изображена схема концепции для реализации веб-приложения.

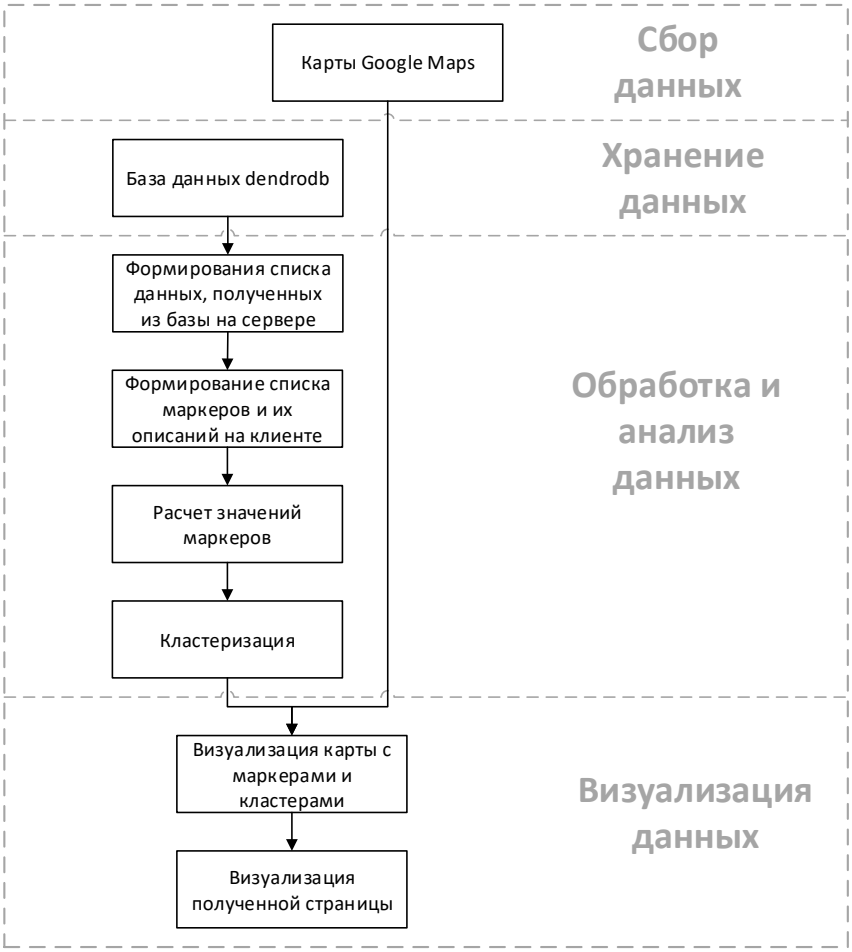

Рисунок 6 – Схема концепции реализации проекта

Данные для проекта будут храниться в базе данных dendrodb. Сначала данные передаются на сервер, который на основе заданной структуры формирует из них список и передает его на клиентскую часть. В ней на основе этих данных формируется список маркеров, определяется их описание и значения. После этого маркеры добавляются в кластер, который будет проводить кластеризацию по заданному алгоритму. В результате на странице сайта отображается карта, полученная из сервиса Google Maps, и на ней отображается список маркеров и кластеров. В конечном результате выводится страница с картой и описанием информации о кластерах.

3.2 Серверная часть

Серверная часть представляет собой REST интерфейс, связывающий сервер с клиентом. При запуске сервера запускается и REST интерфейс, который будет принимать запросы, собирать данные из базы данных и отправлять их в ответ на запрос по протоколу HTTP (HTTPS). Данные REST интерфейс береz из базы данных dendrodb. Данная база состоит из 3 таблиц (dendromesument, dendromarker и dendroimg), изображенных на рисунке 7.

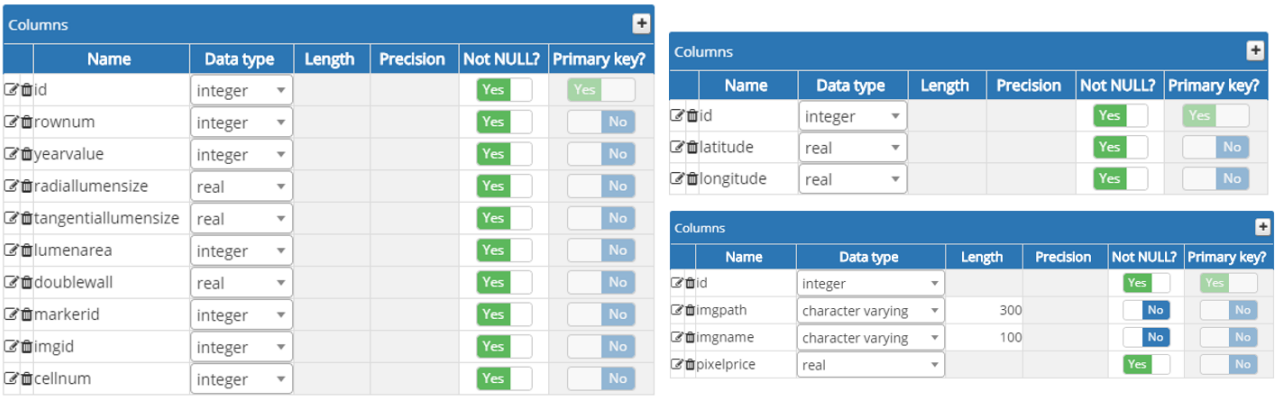

Рисунок 7 – Таблицы базы данных dendrodb

Рисунок 10 – Форма задаваемого года

Форма ограничена годами от 1800 до 2050 и обрабатывается нажатием на кнопку «Enter» при редактировании года или нажатием кнопки «Поиск». При этом данные передаются POST запросом, в котором указывается год, по которому необходимо получить данные.

3.3 Клиентская часть

При обращении по адресу страницы сервер возвращает страницу, состоящую из 4 секций: header, footer, панели с Google Maps и панели таблицы для вывода данных о маркерах. После полной загрузки страницы выполняется запрос на сервер с целью получения данных по срезам деревьев за выбранный год (по стандарту получают данные за 2014 год). После того как данные будут получены формируется массив данных для маркеров, на основе которого будут строиться маркеры Google карт. Список маркеров выводится на панели справа (рис. 11).

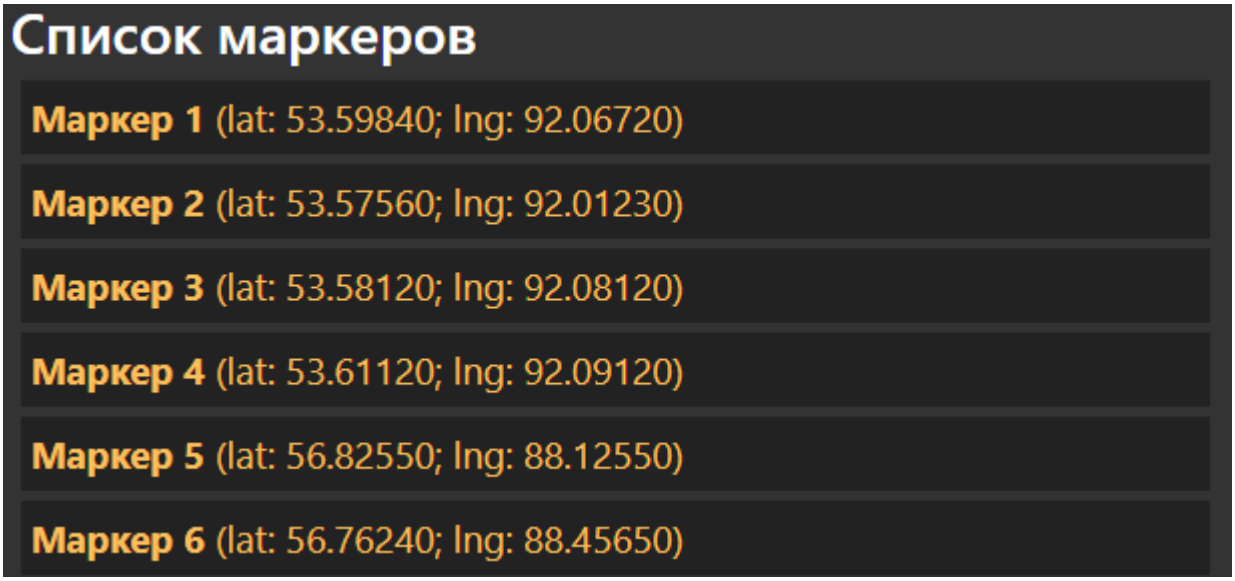

Рисунок 11 – Пример формируемого списка данных

После формирования массива данных на их основе строятся таблицы с информацией о микропараметрах срезов деревьев по заданным координатам. При нажатии на маркер на карте или его значение из списка маркеров откроется информация о нем (рис. 12). Для того, чтобы вернуться к списку маркеров необходимо нажать кнопку «Назад».

Ряд	Клетка	Год	Ширина люмена	Высота люмена	Площадь люмена	Двойная стенка
1	1	2014	60	50	2500	14
1	2	2014	63	54	2654	12
1	3	2014	57	50	2450	14
1	4	2014	58	49	2430	14

Рисунок 12 – Таблица данных о микропараметрах среза дерева

Для получения данных за другой год необходимо выбрать его в форме, изображенной на рисунке 10 и список маркеров автоматически обновится.

Значения маркеров описывают один из четырех параметров: радиальный, тангенциальный размер люмена, площадь люмена или толщину двойной стенки. Для того, чтобы выбрать параметр, по которому будут визуализироваться маркеры, необходимо выбрать его из списка параметров, заданных в выпадающем меню, рядом с формой выбора года (рис. 13).

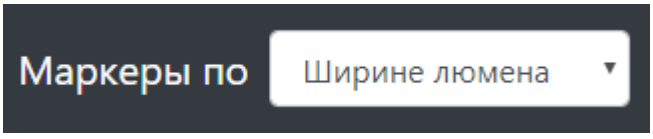

Рисунок 13 – Выпадающее меню выбора значений для маркеров

Маркеры представлены в виде круга, расположенного по заданным координатам, с определяемым значением и полосой вдоль него, визуализирующей значение маркера (рис. 14). Значение полосы и маркера автоматически обновляются из заданного массива маркеров при выборе значений для маркера из выпадающего меню, изображенного на рисунке 13.

Рисунок 14 – Маркеры срезов деревьев на карте Google

Для размеров люмена и стенки достаточно линии с распределением размеров для диапазона от 0 до 100. Данная линия обозначается желтым цветом. Для удобства значения площади люмена с диапазоном значений от 0 до 3000, используется обозначение в виде красной линии (рис. 15).

Рисунок 15 – Маркеры с обозначением средних размеров площади люмена

При нажатии на маркер или выбрав его из списка маркеров формируется всплывающее окно с заданной информацией, например, точные координаты данного маркера (рис. 16).

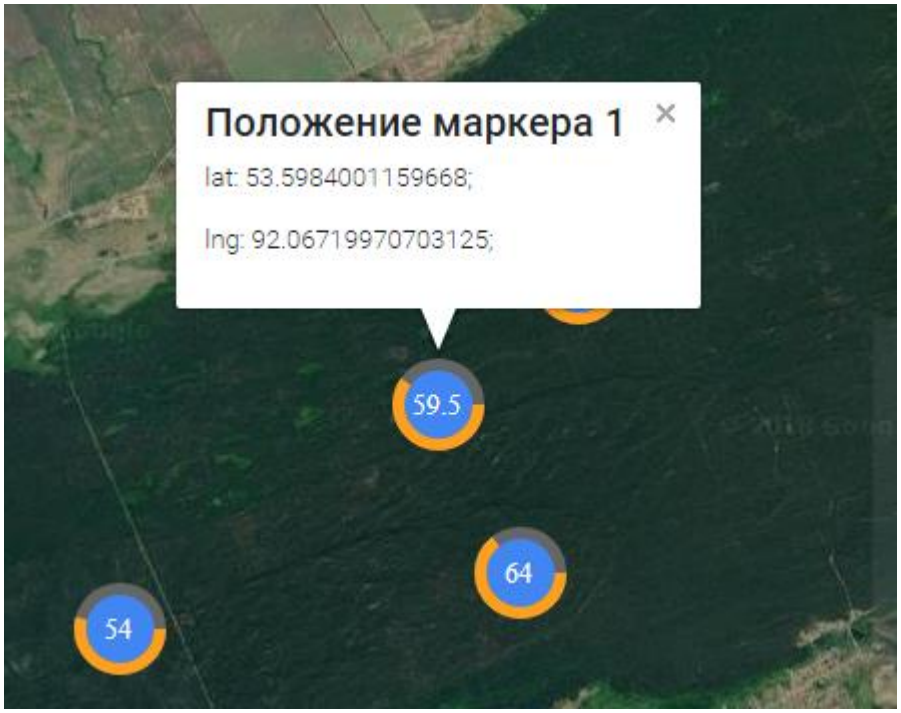

Рисунок 16 – Вывод информации о положении маркера

При уменьшении масштаба маркеры перекрываются друг другом, из-за чего становятся малоразличимыми. Для решения этой проблемы используется кластеризация, позволяющая объединять несколько маркеров в один кластер (рис. 17).

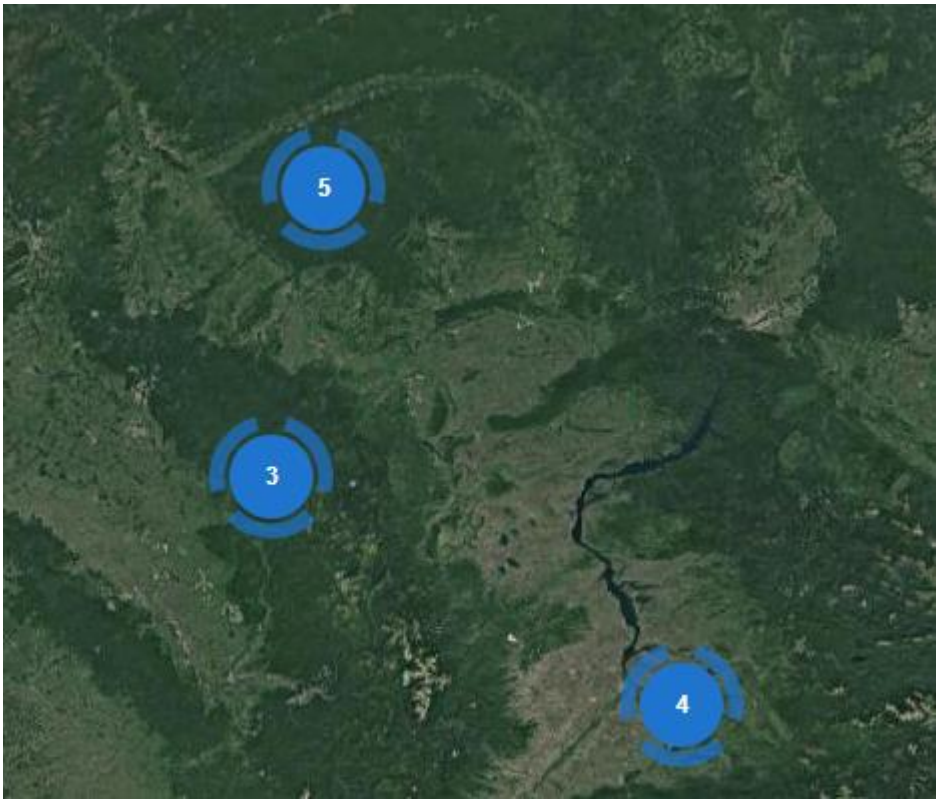

Рисунок 17 – Кластеризация ближайших маркеров

Кластеризация имеет несколько градаций по количеству маркеров внутри себя. Например, для 2-10 маркеров используется синий маркер со стандартным размером маркера кластера, для 10-100 – фиолетовый с размером 1.2 от стандартного и т.д. (рис. 18).

Рисунок 18 – Два вида маркеров для кластеров различных размеров

Для получения информации о маркерах внутри кластера можно нажать на него левой кнопкой мыши (рис. 19). Использование масштабирование до маркера для приближения к маркерам внутри кластера перенесено на комбинацию «CTRL + ЛКМ».

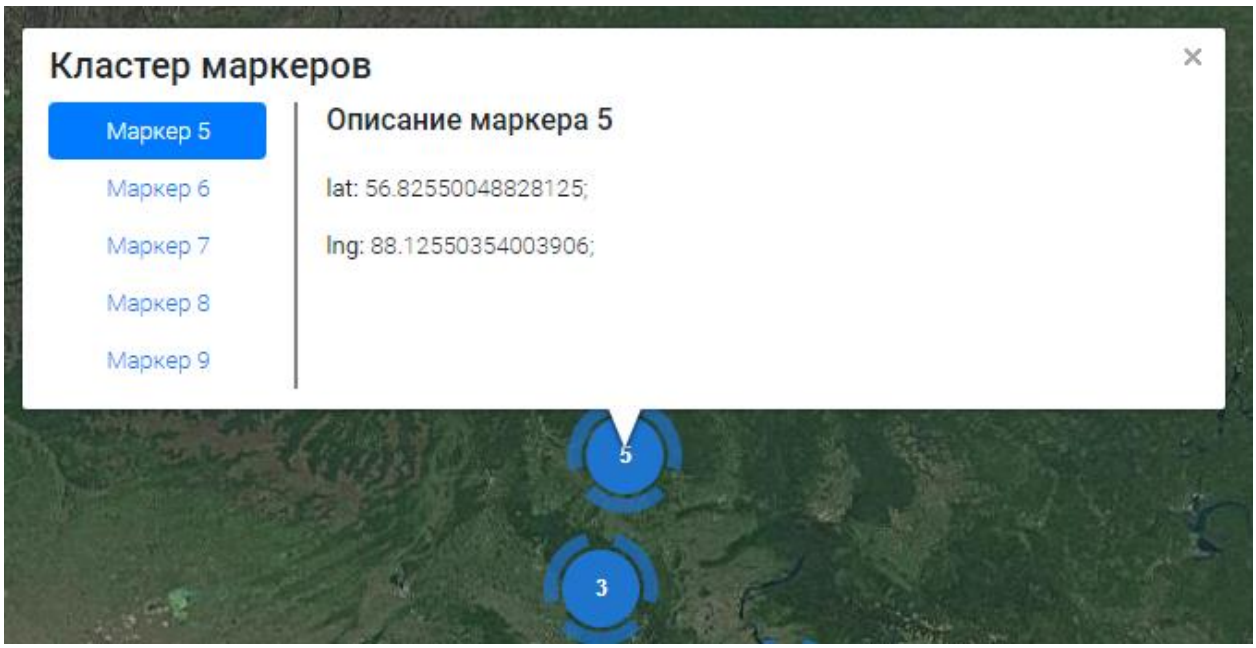

Рисунок 19 – Получение информации о маркерах внутри кластера

Так же реализовано модальное окно для пользователей, содержащее некоторую информацию о сайте, содержащее список основных действий с картой. (рис. 20).

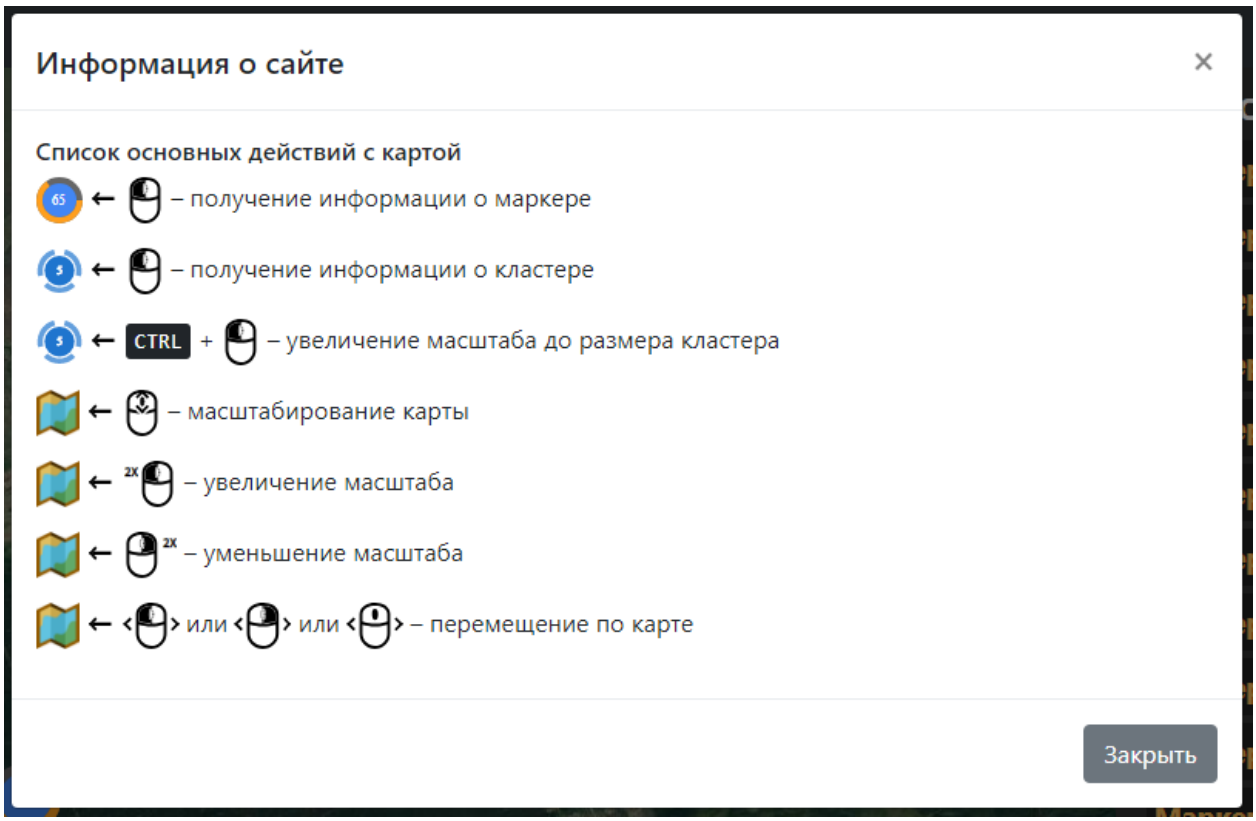

Рисунок 20 – Информация о работе с веб приложением

Данный набор инструментов позволяет полноценно визуализировать информацию о микропараметрах деревьев и предоставляет интерактивность и визуализацию данных на карте и в виде таблицы. Интерфейс максимально упрощен для понятной и удобной работы пользователя с ним, что не составит труда воспользоваться веб приложением даже мало знакомым с данной тематикой людям.

3.4 Руководство пользователя

Для работы с данным веб-приложением клиент должен иметь одну из поддерживаем операционных систем, таких как Windows 10, MacOS или Unix-подобные операционные системы последних версий. Так же необходим браузер, поддерживающий работу современных веб-сервисов. К таким относится большинство браузеров последних версий, за исключением IE.

Чтобы начать работу с веб-сервисом пользователю необходимо ввсести в адресную строку URL используемого сервиса, если он установлен на удаленном хосте, или «localhost:port», где порт указывается в программе при запуска сервиса (по стандарту – 3000) (рис. 21).

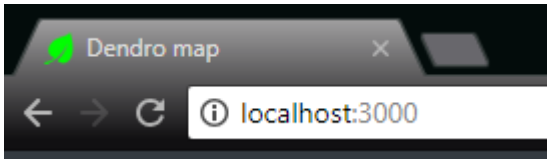

Рисунок 21 – Ввод URL локального сервера

После ввода URL пользователю откроется сайт, содержащий заголовок сайта, панель с Google картой, списком маркером и подвала сайта. Заголовок веб-приложения содержит названия сайта, кнопку получения информации, выпадающее меню с выбором отображения значений маркера и выбором года (рис. 22).

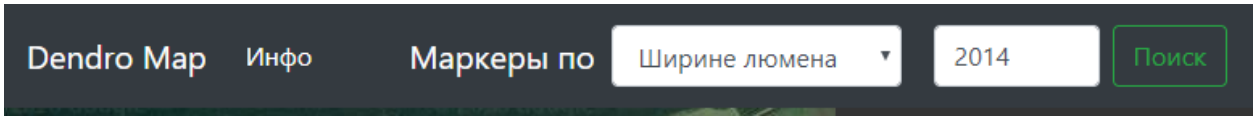

Рисунок 22 – Заголовок сайта

При нажатии на кнопку инфо выведется информация о работе с картой, как это было изображено на рисунке 20. Выбрав значение в выпадающем меню «Маркеры по» изменится выводимое значение маркеров в зависимости от выбранного значения (рис. 23). Последняя часть отвечает за выбор года, по которому будут выводиться маркеры.

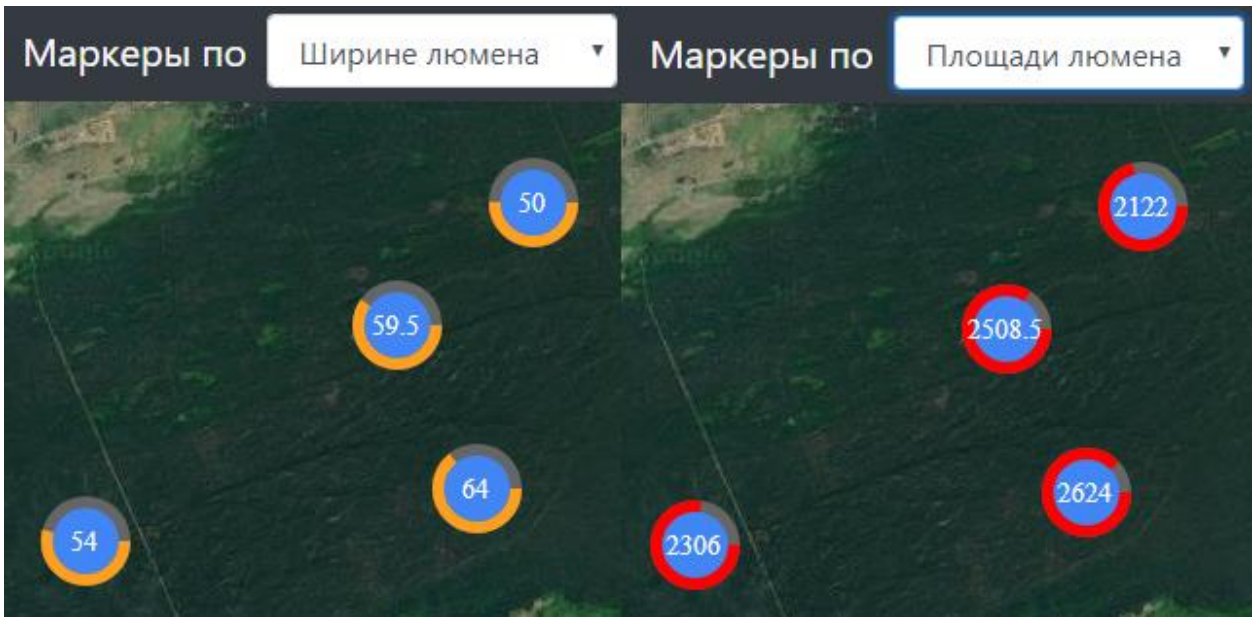

Рисунок 23 – Изменение значений маркера при изменении значения в выпадающем меню «Маркеры по»

Панель с Google картой отображает маркеры за выбранный год. Каждый маркер обозначает координаты среза дерева, по которому проводились расчеты. Чтобы получить информацию о маркере необходимо нажать на него левой кнопкой мыши, после чего выведется информация о точном местоположении маркера и таблица значений среза дерева (рис. 24). Так же можно выбрать нужный маркер из списка маркеров изображенного на рисунке 11.

Ряд	Клетка	Год	Ширина люмена	Высота люмена	Площадь люмена	Двойная стенка
1	1	2014	60	50	2500	14
1	2	2014	63	54	2654	12
1	3	2014	57	50	2450	14
1	4	2014	58	49	2430	14

Рисунок 24 – Вывод информации о маркере

Чтобы закрыть информацию о маркере можно нажать на крестик в правой верхней части информационного окна маркера, нажатием левой кнопки мыши на свободное пространство в области карты или нажатием кнопки «Назад» в панели с таблицей значений маркера.

Таблицу значений можно сортировать по каждому из столбцов по убыванию или возрастанию. Пример сортировки таблицы по возрастанию значений столбца «Ширина люмена» представлена на рисунке 25.

Ряд	Клетка	Год	Ширина люмена	Высота люмена	Площадь люмена	Двойная стенка
1	3	2014	57	50	2450	14
1	4	2014	58	49	2430	14
1	1	2014	60	50	2500	14
1	2	2014	63	54	2654	12

Рисунок 25 – Пример сортировки таблицы по столбцу «Ширина люмена»

Перемещаться по карте можно перемещением мыши, предварительно зажав любую из трех ее основных кнопок.

Для изменения масштаба карты можно воспользоваться колесиком мыши, двойным щелчком левой или правой кнопки мыши или кнопками изменения масштаба в правом нижнем углу Google карт (рис. 26).

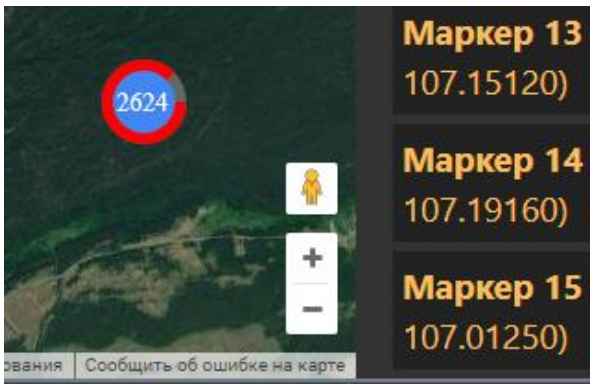

Рисунок 26 – Кнопки изменения масштаба карты

При изменении масштаба рядом стоящие маркеры объединятся в один кластер, как это показано на рисунке 17. Чтобы посмотреть информацию о кластере необходимо нажать на него левой кнопкой мыши (рис. 19). Чтобы увидеть маркеры внутри кластера – необходимо увеличить масштаб карты или нажать «CTRL + ЛКМ» по кластеру для увеличения масштаба карты до размера кластера.

Для отображений названий объектов на карте необходимо навести курсок на кнопку «Спутник» и нажать на появившуюся кнопку «Названия объектов» (рис. 27).

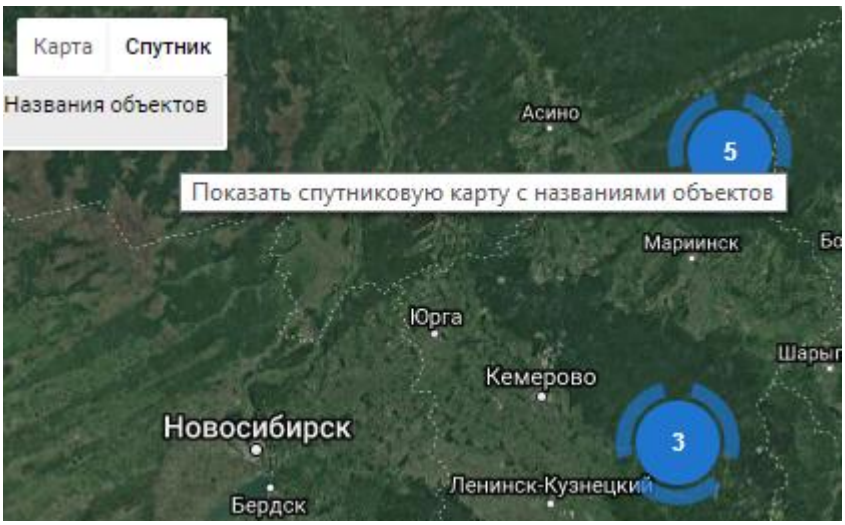

Рисунок 27 – Отображение объектов на карте

Чтобы изменить вид карты необходимо нажать кнопку «Карта», радом с кнопкой «Спутник» (рис. 28).

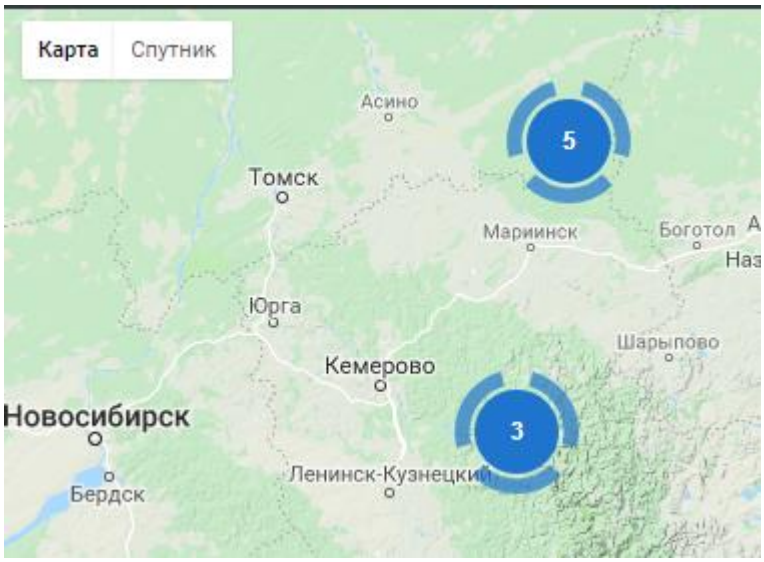

Рисунок 28 – Отображение иного вида карты

Для работы с картой в полноэкранном режиме необходимо нажать на кропку в правой верхней части карты (рис. 29).

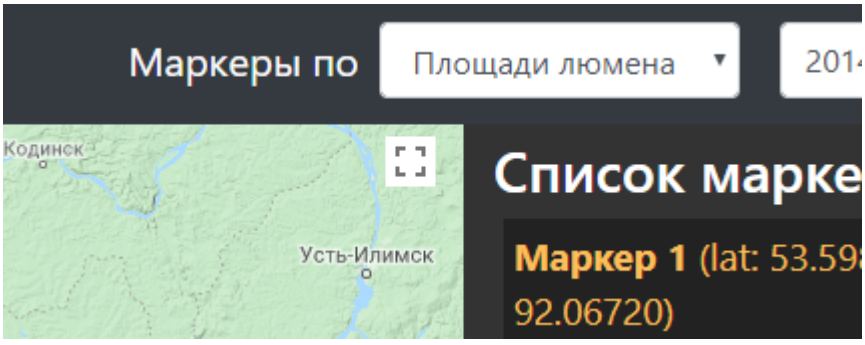

Рисунок 29 – Кнопка полноэкранного режима карты

Для закрытия веб приложения достаточно закрыть вкладку в браузере или сам браузер. Таким образом выполняется работа с веб-приложением, визуализирующим информацию о срезах деревьев.

3.5 Выводы по главе 3

В данной главе был описан процесс разработки серверной и клиентской части приложения, руководство пользователя. Был подробно расписан процесс получения данных из базы данных, преобразование данных для передачи их на клиента, обработки данных на клиенте и визуализации этих данных. В результате было получено веб-приложение, получающее данных из базы данных и визуализирующее его на карте сервиса Google Maps.

ЗАКЛЮЧЕНИЕ

В ходе проведения анализа методов картографирования были выбраны наиболее подходящие для построения приложения, способного визуализировать информацию о срезах деревьев. Был проведен анализ получаемых параметров из срезов деревьев и выявлена основная информация, которая будет храниться в базе данных. Обзор веб-картографических сервисов дал определения API для визуализации данных на карте, а обзор языков позволил определиться с инструментами для реализации серверной части.

Определившись с инструментами, фреймворками и библиотеками для разработки веб-приложения была описана схема концепции для реализации работы. Основываясь на ней были сформирован прототип разрабатываемого веб-приложения.

В результате выпускной квалификационной работы было разработано веб-приложение, состоящее из клиентской и серверной части, соединенных между собой через REST интерфейс. Сервер хранит данные в базе данных PostgreSQL, получает их и передает при помощи возможностей языка Go. Клиентская часть оформлена при помощи Фреймворка Bootstrap 4, и предоставляет интерактивный пользовательский интерфейс, что позволяет пользователю быстрее адаптироваться к данному приложению.

Цели, поставленные в выпускной квалификационной работе, были выполнены полностью, в результате чего было реализовано веб-приложение, выполняющее все необходимые задачи.

СПИСОК ИСПОЛЬЗОВАННЫХ СОКРАЩЕНИЙ

IDE – интегрированная среда разработки

GUI – графический пользовательский интерфейс

API – программный интерфейс приложения

СУБД – система управления базой данных

ГИС – геоинформационная система

БД – база данных

WMTS – веб-карта

IE – Internet Explorer

OSM – OpenStreetMap

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шишов, В. В. Визуализация и анализ дендроклиматической информации на основе интерактивной системы дендроклиматического мониторинга : автореф. дис. … д-ра тех. наук : 05.13.01 / Шишов Владимир Валерьевич. – Братск, 2009 – 300 с.

2. Баранов, Ю. Б. Геоинформатика. Толковый словарь основных терминов / под. ред. А. М. Берлянта, А. В. Кошкарева. – Москва : ГИС-Ассоциация, 1999. – 205 с.

3. Быков, А. В. Web-картографирование : учеб. пособие / А. В. Быков, С. В. Пьянков ; перм. гос. нац. исслед. ун-т. – Пермь, 2015. – 110 с.

4. ArcGIS Enterprise [Электронный ресурс] : Картографические сервисы. – Режим доступа : http://enterprise.arcgis.com/ru/server/latest/publish-services/linux/what-is-a-map-service.htm

5. Назаров, Р. Р. Сбор и обработка данных дистанционного зондирования : учеб. пособие / Р. Р. Назаров. – Казань : Казан. Ун-т, 2015. – 62 с.

6. Савельев, А. С. Проектирование геоинформационных систем : учеб. Пособие / А. С. Савельев, А. А. Гостева. – Красноярск : Сибирский федеральный университет, 2010. – 176 с.

7. Шишов, В. В. Методы анализа дендроклиматических данных и их применения для территории Сибири : учеб. пособие / В. В. Шишов, И. И. Тычков, М. И. Попкова. ФГАОУ ВПО «Сибирский федеральный университет». – Красноярск, 2015 – 211 с.

8. Ваганов, Е. А. Динамика роста годичных колец хвойных пород. Отражение условий среды прошлого и будущего. Экологические исследования / Е. А. Ваганов // Лесоведение / Наука ; Москва, 2007 – 70-71 с.

9. Федоров, П. П. Дендроклиматический анализ радиального прироста деревьев в Центральной Якутии : дис. … канд. с.-х. наук : 03.00.16 / Федоров Павел Петрович. – Братск, 2008. – 151 с.

10. Ваганов, Е. А. Дендрохронология : учеб. пособие / Е. А. Ваганов, В. Б. Круглов, В. Г. Васильев. – Красноярск : ФГАОУ ВПО «Сибирский федеральный университет», 2008. - 109 с.

ПРИЛОЖЕНИЕ А

Исходный код

main.go

```
package main

import (
   "fmt"
   "net/http"
   "html/template"
   "github.com/jmoiron/sqlx"
   "encoding/json"
   _ "github.com/lib/pq"
   "strconv"
)

var (
   err error
   db *sqlx.DB
)

// структура для получения данных из БД
type Mes struct {
   ID                int       `db:"id"`
   CellNum            int       `db:"cellnum"`
   RowNum          int       `db:"rownum"`
   YearValue        int       `db:"yearvalue"`
   RadialLumenSize       float64    `db:"radiallumensize"`
   TangentialLumenSize    float64    `db:"tangentiallumensize"`
   LumenArea        float64 `db:"lumenarea"`
   DoubleWall       float64    `db:"doublewall"`
   MarkerID         int       `db:"markerid"`
   ImgID            int       `db:"imgid"`
   Latitude         float64    `db:"latitude"`
   Longitude        float64 `db:"longitude"`
}

type getDataStruct struct {
   Year   int       `json:"year"`
}

func indexHandler(w http.ResponseWriter, r *http.Request) {
   t, err := template.ParseFiles("templates/index.html",
"templates/header.html", "templates/footer.html")
   if err != nil {
      fmt.Fprintf(w, err.Error())
   }

   t.ExecuteTemplate(w, "index", nil)
```

```
}

func getData(w http.ResponseWriter, r *http.Request) {
   err = r.ParseForm()
   PanicOnError(err)
   val := r.Form.Get("year")
   year, err := strconv.Atoi(val)
   PanicOnError(err)
   rows, err := db.Queryx(`SELECT dms.*, dm.latitude,
dm.longitude
                           FROM dendromeasurement dms INNER JOIN
dendromarker dm
                           ON dms.markerid = dm.id WHERE
dms.yearvalue = $1`, year)
   PanicOnError(err)
   defer rows.Close()
   arr := make([]Mes, 0)
   var data Mes
   for rows.Next() {
      err = rows.StructScan(&data)
      PanicOnError(err)
      arr = append(arr, data)
   }
   j, err := json.Marshal(&arr)
   w.WriteHeader(http.StatusOK)
   w.Write(j)
}

func main() {
   // Подключение базы
   db, err = sqlx.Connect("postgres", "user=postgres
password=23256616snm dbname=dendrodb sslmode=disable")
   PanicOnError(err)
   // Проверяем соединение с базой
   err = db.Ping()
   PanicOnError(err)

   // референсы
   http.Handle("/assets/", http.StripPrefix("/assets/",
http.FileServer(http.Dir("./assets/"))))
   http.HandleFunc("/", indexHandler)
   http.HandleFunc("/getdata", getData)

   // слушать порт 3000
   http.ListenAndServe(":3000", nil)
}

func PanicOnError(err error) {
   if err != nil {
      panic(err)
   }
}
```

header.html

```
{{ define "header" }}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <link rel="icon" type="image/png" href="assets/content/favicon.png"
/>
    <link rel="stylesheet" href="assets/css/bootstrap.min.css">
    <link rel="stylesheet" href="assets/css/main.css">
    <script src="assets/js/jquery-3.3.1.min.js"></script>
    <script src="assets/js/markerclusterer.js"></script>
    <script src="assets/js/main.js"></script>
</head>
<body>

    <header>
        <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-
dark">
            <!-- Лого -->
            <a class="navbar-brand" href="#">Dendro Map</a>
            <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarCollapse" aria-
controls="navbarCollapse" aria-expanded="false" aria-label="Toggle
navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarCollapse">
                <!-- Информация о приложении -->
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item active">
                        <a class="nav-link" href="#" data-toggle="modal"
data-target="#infoModal">Инфо<span class="sr-
only">(current)</span></a>
                    </li>
                </ul>
                <!-- Поиск по году -->
                <form class="form-inline mt-2 mt-md-0"
onsubmit="GetDataByYear(document.getElementById('searchYear').value);
return false;">
                    <label for="markerVisType" class="col-form-label-lg"
style="padding-right: 10px; color:white">Маркеры по </label>
                    <select onchange="changeMarkerVT()" class="form-control
form-control" id="markerVisType" style="margin-right: 20px">
                        <option value=0>Ширине люмена</option>
                        <option value=1>Высоте люмена</option>
                        <option value=2>Площади люмена</option>
                        <option value=3>Двойной стенке</option>
                    </select>
                    <input class="form-control mr-sm-2" type="number"
min="1800" max="2050" id="searchYear" value=2014 placeholder="Год..."
aria-label="Год">
                    <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Поиск</button>
```

```
            </form>
          </div>
        </nav>
    </header>

    <!-- Modal -->
    <div class="modal fade" id="infoModal" tabindex="-1" role="dialog"
aria-labelledby="exampleModalLabel" aria-hidden="true">
      <div class="modal-dialog modal-lg" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="exampleModalLabel">Информация о
сайте</h5>
            <button type="button" class="close" data-dismiss="modal"
aria-label="Close">
              <span aria-hidden="true">&times;</span>
            </button>
          </div>
          <div class="modal-body">
            <h6>Список основных действий с картой</h6>
            <p><img src="assets/content/markerinfo.svg" height="30">
<img src="assets/content/left_cur.svg" height="30"> <img
src="assets/content/lmb.svg" height="30"> &ndash; получение информации
о маркере</p>
            <p><img src="assets/content/clust.svg" height="30"> <img
src="assets/content/left_cur.svg" height="30"> <img
src="assets/content/lmb.svg" height="30"> &ndash; получение информации
о кластере</p>
            <p><img src="assets/content/clust.svg" height="30"> <img
src="assets/content/left_cur.svg" height="30"> <kbd>CTRL</kbd> + <img
src="assets/content/lmb.svg" height="30"> &ndash; увеличение масштаба
до размера кластера</p>
            <p><img src="assets/content/map.png" height="30"> <img
src="assets/content/left_cur.svg" height="30"> <img
src="assets/content/scroll.svg" height="30"> &ndash; масштабирование
карты</p>
            <p><img src="assets/content/map.png" height="30"> <img
src="assets/content/left_cur.svg" height="30"> <img
src="assets/content/2x.svg" height="30"><img
src="assets/content/lmb.svg" height="30"> &ndash; увеличение
масштаба</p>
            <p><img src="assets/content/map.png" height="30"> <img
src="assets/content/left_cur.svg" height="30"> <img
src="assets/content/rmb.svg" height="30"><img
src="assets/content/2x.svg" height="30"> &ndash; уменьшение
масштаба</p>
            <p><img src="assets/content/map.png" height="30"> <img
src="assets/content/left_cur.svg" height="30">
              <img src="assets/content/left_nar.svg" height="30"><img
src="assets/content/lmb.svg" height="30"><img
src="assets/content/right_nar.svg" height="30"> или
              <img src="assets/content/left_nar.svg" height="30"><img
src="assets/content/rmb.svg" height="30"><img
src="assets/content/right_nar.svg" height="30"> или
              <img src="assets/content/left_nar.svg" height="30"><img
src="assets/content/midmb.svg" height="30"><img
src="assets/content/right_nar.svg" height="30">
            &ndash; перемещение по карте</p>
```

```
          </div>
          <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-
dismiss="modal">Закрыть</button>
          </div>
        </div>
      </div>
    </div>

{{ end}}
```

index.html

```
{{ define "index" }}

{{ template "header" }}


<main class="container-fluid" role="main">

      <div class="row">
        <!-- Карта | список/описание маркеров -->
        <div class="col-8" id="map"></div>
        <div class="col-4" id="table">
            <!-- Список маркеров -->
            <div id='markersListDiv' style="padding: 10px 0 0
0;">
                <h3 style="color: white">Список маркеров</h3>
                <ul id='markersList'></ul>
            </div>
            <div id="BackToTM" class="mdescr row"
style="padding: 10px; display: none">
                <button id="BTTButton" class="col-3 btn"
onclick="BackToTableMarkers()">◄ Назад</button>
                <h3 class="col-9" style="color: white">Таблица
значений маркера</h3>
            </div>
            <!-- Таблица микпропараметров клеток деревьев -->
            <table id="markerTable" class="table table-dark
table-bordered table-hover" style="display: none">
                <thead>
                    <tr>
                        <th onclick="sortTable(0)">Ряд</th>
                        <th onclick="sortTable(1)">Клетка</th>
                        <th onclick="sortTable(2)">Год</th>
                        <th onclick="sortTable(3)">Ширина
люмена</th>
                        <th onclick="sortTable(4)">Высота
люмена</th>
                        <th onclick="sortTable(5)">Площадь
люмена</th>
                        <th onclick="sortTable(6)">Двойная
стенка</th>
                    </tr>
```

```
                </thead>
                <tbody>

                </tbody>
            </table>
        </div>

    </div>

</main>

{{ template "footer" }}

{{ end }}
```

footer.html

```
{{ define "footer" }}

<footer>
   Мамышева Г.А. КИ14-16Б
</footer>

<script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBEBXjwMiB
PJS5LdYIc4ESGkPmnburLyfo&callback=initMap" async defer></script>
<script src="assets/js/bootstrap.min.js"></script>

</body>
</html>

{{ end }}
```

main.css

```
#map {
  min-height: 875px;
}
#table {
  background-color: #333;
}

main {
  margin-top: 18px;
  margin-right: 0;
}
header {
  padding: 10px;
  height: 40px;
}
header h1 {
  float:left;
```

```
  color: white;
  font-size: 24pt;
  margin: 0;
  padding: 0;
}
footer {
  clear: both;
  background-color: black;
  color: white;
  padding: 5px;
  text-align: center;
  margin-top:2px;
}
/* Optional: Makes the sample page fill the window. */
html, body {
  height: 100%;
  margin: 0;
  padding: 0;
  background-color: #444;
}
main {
  border-top: 2px slategray solid;
  border-bottom: 2px slategray solid;
}

#markersList {
  list-style: none;
  font-size: 14pt;
  padding: 0;
  margin: 0;
}
.MLNode {
  text-decoration: none;
  color: #fcbf5d;
  margin: 5px;
  padding: 5px;
  background: #222;
  cursor: pointer;
}
.MLNode:hover {
  color: #cc670a;
}
#markerTable th {
  cursor: pointer;
  -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
}
#markerTable th:hover {
  color: orange;
  background: #444
}
```

main.js

```
var map;
var markerCluster;
var markers = {};
var mapMrkrs = [];
var infowindow;
var popupCnt;

// marker.setIcon(getCircle(100));

function getSVGMarker(col, val, type) {
    var color = "#FF9F1E"
    if (type != 1)
        color = "#FF0000"
    return "<?xml version='1.0'?><svg width='46' height='46'
viewBox='0 0 46 46' version='1.1'
xmlns='http://www.w3.org/2000/svg'>" +
                "<circle r='20' cx='23' cy='23' fill='#4286f4'
stroke-dasharray='125' stroke-dashoffset='0' stroke='#666'
stroke-width='6'></circle>" +
                "<circle class='a" + new Date().getTime() + "'
r='20' cx='23' cy='23' fill='transparent' stroke-
dasharray='125.5' stroke-dashoffset='" + col + "' stroke='" +
color + "' stroke-width='6'>" +
                "<animate attributeName='stroke-dashoffset'
from='125.5' to='" + col + "' dur='1s' /></circle>" +
                "<text x = '22' y = '28'  font-size='14' text-
anchor='middle' fill='white'>" + val + "</text>" +
            "</svg>";
}

function changeMarkerVT() {
    var valID = $("#markerVisType").val();
    for (var i = 0; i < mapMrkrs.length; i++ ) {
        mID = mapMrkrs[i].title;
        var avg;
        switch (valID) {
            case "0":
                avg = markers[mID].avgRLu();
                break
            case "1":
                avg = markers[mID].avgTLu();
                break
            case "2":
                avg = markers[mID].avgLuArea();
                break
            case "3":
                avg = markers[mID].avgDW();
                break
        }
```

```
            mapMrkrs[i].setIcon(getCircle(avg));
        }

    }


    // Всплывающее окно
    class popupContent {
        constructor() {
            this.Title = ""
        }
        setPopup(markerID) {
            infowindow.setContent("<h5>Положение маркера " +
    markerID + "</h5>" +
                "<p>lat: " + markers[markerID].lat + ";</p>" +
                "<p>lng: " + markers[markerID].lng + ";</p>");
        }
        setClustPopup(markerIDs) {
            var res = "<h5>Кластер маркеров</h5>";
            var s1 = "";
            var s2 = "";
            var b1 = " active";
            var b2 = " show active"
            markerIDs.forEach(function(item, i) {
                if (i > 0) {
                    b1 = "";
                    b2 = "";
                }
                var el = markers[item];
                s1 += "<a class='nav-link .clust-tab " + b1 + "'
    data-toggle='pill' href='#marker" + item + "' role='tab' aria-
    controls='clust' aria-selected='true'>Маркер " + item + "</a>";
                s2 += "<div class='tab-pane fade" + b2 + "'
    id='marker" + item + "' role='tabpanel' aria-labelledby='clust-
    tab'>" +
                "<h6 style='margin-bottom:20px'>Описание маркера " +
    item + "</h6><p><strong>lat:</strong> " + el.lat +
    ";</p><p><strong>lng:</strong> " + el.lng + ";</div>";
            });
            res = "<h5 style='width:600px'>Кластер маркеров</h5>" +
            "<div class='container-fluid row' style='padding:0;
    margin-left: -15px'>" +
                "<div class='col-3' style='text-align: center;
    border-right: 2px solid gray'>" +
                    "<div class='nav flex-column nav-pills'
    id='clusts-tab' role='tablist' aria-orientation='vertical'>" +
    s1 +
                    "</div>" +
                "</div>" +
                "<div class='col-9'>" +
                    "<div class='tab-content' id='clust-
    tabContent'>" + s2 +
                    "</div>" +
```

```
                "</div>" +
            "</div>";
            infowindow.setContent(res);
        }
    }
    // класс измерений для маркера
    class Msmnt {
        constructor(item) {
            this.ID = item.ID;
            this.cellNum = item.CellNum;
            this.rowNum = item.RowNum;
            this.yearValue = item.YearValue;
            this.RLu = item.RadialLumenSize;
            this.TLu = item.TangentialLumenSize;
            this.LuArea = item.LumenArea;
            this.DW = item.DoubleWall;
            this.imgID = item.ImgID;
        }
    }
    // Класс маркеров
    class Mrk {
        constructor(item) {
            this.markerID = item.MarkerID;
            this.lat = item.Latitude;
            this.lng = item.Longitude;
            this.Mes = []
            this.addMes(item)
        }
        // добавление измерения
        addMes(item) {
            var msmnt = new Msmnt(item)
            this.Mes.push(msmnt)
        }
        avgRLu() {
            var res = 0;
            var count = 0;
            this.Mes.forEach(function (item) {
                res += item.RLu;
                count++;
            })
            return res/count;
        }
        avgTLu() {
            var res = 0;
            var count = 0;
            this.Mes.forEach(function (item) {
                res += item.TLu;
                count++;
            })
            return res/count;
        }
        avgDW() {
            var res = 0;
```

```
        var count = 0;
        this.Mes.forEach(function (item) {
            res += item.DW;
            count++;
        })
        return res/count;
    }
    avgLuArea() {
        var res = 0;
        var count = 0;
        this.Mes.forEach(function (item) {
            res += item.LuArea;
            count++;
        })
        return res/count;
    }
    // получение положение маркера
    pos() {
        return {lat: this.lat, lng: this.lng}
    }
}

// получение данных при полной загрузке страницы
window.onload = function() {
    Clast()
    GetDataByYear(2014); // изначально для 2014 года
    popupCnt = new popupContent();  // создаем popup окно
    infowindow = new google.maps.InfoWindow({
        content: "" // сначала никакой информации в popup окне
    });
    google.maps.event.addListener(infowindow, 'closeclick',
function() {
        BackToTableMarkers();
    });
    google.maps.event.addListener(map, 'click', function() {
        infowindow.close();
        BackToTableMarkers();
    });
}

// Создания карты
function initMap() {
  map = new google.maps.Map(document.getElementById('map'), {
    center: {lat: 53.58839, lng: 92.03717},
    zoom: 12,
    mapTypeId: 'satellite'
  });
}

// Получение данных за год
function GetDataByYear(year) {
    for (var i = 0; i < mapMrkrs.length; i++ ) {
        mapMrkrs[i].setMap(null);
```

```
    }
    markerCluster.clearMarkers();
    mapMrkrs.length = 0;
    markers = {};
    $.post( "/getdata", {'year' : year } )
        .done(function(data) {
            arr = JSON.parse(data)
            // Заполнение массива данными
            arr.forEach(function(item) {
                if (markers[item.MarkerID] == undefined) {
                    markers[item.MarkerID] = new Mrk(item)
                } else {
                    markers[item.MarkerID].addMes(item)
                }
            });
            var avg = ""
            // Формирование маркеров
            for (var key in markers) {
                avg = (markers[key].avgRLu()).toString()
                var marker = new google.maps.Marker({
                    position: markers[key].pos(),
                    map: map,
                    title: key,
                    optimized: false,
                    icon: getCircle(avg)
                });

                mapMrkrs.push(marker)
            }
            mapMrkrs.forEach(function(marker) {
                google.maps.event.addListener(marker, 'click',
function() {
                    popupCnt.setPopup(Number(marker.title));
                    infowindow.open(map, marker);

marker.setAnimation(google.maps.Animation.BOUNCE);
                    setTimeout(function () {
                        marker.setAnimation(null);
                    }, 500)
                    CreateMesTable(Number(marker.title));
                });
            });
            markerCluster.addMarkers(mapMrkrs);
            CreateListOfMarkers();
        })
        .fail(function() {
            alert( "Error!" );
        });
}

// создать таблицу измерений
function CreateMesTable(id) {
```

```
document.getElementById("markerTable").childNodes[3].innerHTML =
""
    $("#markersListDiv").fadeOut();
    mt = $("#markerTable");
    setTimeout(function () {
        $("#BackToTM").fadeIn();
        mt.fadeIn();
    }, 500);
    markers[id].Mes.forEach(function (item) {
        mt.append("<tr><td>" + item.rowNum + "</td><td>" +
item.cellNum + "</td><td>" + item.yearValue + "</td><td>" +
item.RLu + "</td><td>" + item.TLu + "</td><td>" + item.LuArea +
"</td><td>" + item.DW + "</td></tr>");
    });
    sortTable(1);
}

// создать список маркеров (измерений)
function CreateListOfMarkers() {
    $("#BackToTM").fadeOut();
    $("#markerTable").fadeOut();
    document.getElementById("markersList").innerHTML = "";
    ml = $("#markersList")
    var i = 0
    for (var key in markers) {
        ml.append("<li onclick='linkClick("+ i +");'
class=\"MLNode\"><b>Маркер " + key + " </b>(lat: " +
parseFloat(markers[key].lat).toFixed(5) + "; lng: " +
parseFloat(markers[key].lng).toFixed(5) +
        ")</li>");
        i++;
    }
    ml.fadeIn();
}

// обработка нажатия на ссылку
function linkClick(id) {
    google.maps.event.trigger(mapMrkrs[id], 'click');
}

// вернуться к списку маркеров
function BackToTableMarkers() {
    infowindow.close();
    $("#BackToTM").fadeOut();
    $("#markerTable").fadeOut();
    setTimeout(function () {
        $("#markersListDiv").fadeIn();
    }, 500);
}


function getCircle(val) {
    var rval = val;
```

```
    var type = 1;
    if (rval > 100) {
        rval /= 30;
        type = 2;
    }
    var col = ((100.0-rval)/100.0)*125.5
    var svg = getSVGMarker(col, val, type);
    return { url: '' +
btoa(svg)
            }
}
// сортировка таблицы
function sortTable(n) {
    var table, rows, switching, i, x, y, shouldSwitch, dir,
switchcount = 0;
    table = document.getElementById("markerTable");
    switching = true;
    dir = "asc";
    while (switching) {
        switching = false;
        rows = table.getElementsByTagName("TR");
        for (i = 1; i < (rows.length - 1); i++) {
            shouldSwitch = false;
            x = rows[i].getElementsByTagName("TD")[n];
            y = rows[i + 1].getElementsByTagName("TD")[n];
            if (dir == "asc") {
                if (x.innerHTML.toLowerCase() >
y.innerHTML.toLowerCase()) {
                    shouldSwitch= true;
                    break;
                }
            } else if (dir == "desc") {
                if (x.innerHTML.toLowerCase() <
y.innerHTML.toLowerCase()) {
                    shouldSwitch = true;
                    break;
                }
            }
        }
        if (shouldSwitch) {
            rows[i].parentNode.insertBefore(rows[i + 1],
rows[i]);
            switching = true;
            switchcount ++;
        } else {
            if (switchcount == 0 && dir == "asc") {
                dir = "desc";
                switching = true;
            }
        }
    }
}
```

```
// задать кластер с заданным цветом
var getGoogleClusterInlineSvg = function (color) {
    var encoded = window.btoa('<svg
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="-100 -100
200 200"><defs><g id="a" transform="rotate(45)"><path d="M0
47A47 47 0 0 0 47 0L62 0A62 62 0 0 1 0 62Z" fill-
opacity="0.7"/></g></defs><g fill="' + color + '"><circle
r="42"/><use xlink:href="#a"/><g transform="rotate(120)"><use
xlink:href="#a"/></g><g transform="rotate(240)"><use
xlink:href="#a"/></g></g></svg>');

    return ('' + encoded);
};
// стили кластеров
var clusterStyles = [
        {
            width: 100,
            height: 100,
            url: getGoogleClusterInlineSvg('#1F75CE'),
            textColor: 'white',
            textSize: 12
        },
        {
            width: 120,
            height: 120,
            url: getGoogleClusterInlineSvg('violet'),
            textColor: 'white',
            textSize: 17
        },
        {
            width: 140,
            height: 140,
            url: getGoogleClusterInlineSvg('green'),
            textColor: 'white',
            textSize: 22
        },
        {
            width: 160,
            height: 160,
            url: getGoogleClusterInlineSvg('orange'),
            textColor: 'white',
            textSize: 27
        },
        {
            width: 180,
            height: 180,
            url: getGoogleClusterInlineSvg('red'),
            textColor: 'white',
            textSize: 32
```

```
        }
    ];
// Кластеризация
function Clast() {
    markerCluster = new MarkerClusterer(map, mapMrkrs,
        {imagePath:
'https://developers.google.com/maps/documentation/javascript/exa
mples/markerclusterer/m',
        styles: clusterStyles,
        zoomOnClick: false
    });
    // действия при нажатии на кластер
    google.maps.event.addListener(markerCluster, 'clusterclick',
function(clust) {
        if (ctrlPressed) {
            var theBounds;
            var mz;
            var mc = clust.getMarkerClusterer();
            // Zoom into the cluster.
            mz = mc.getMaxZoom();
            theBounds = clust.getBounds();
            mc.getMap().fitBounds(theBounds);
            // There is a fix for Issue 170 here:
            setTimeout(function () {
              mc.getMap().fitBounds(theBounds);
              // Don't zoom beyond the max zoom level
              if (mz !== null && (mc.getMap().getZoom() > mz)) {
                mc.getMap().setZoom(mz + 1);
              }
            }, 100);
        } else {
            var cArr = [];
            for (var i = 0; i < clust.getMarkers().length; i++)
{
                cArr.push(clust.getMarkers()[i].title)
        }
        popupCnt.setClustPopup(cArr);
        infowindow.setPosition(clust.getCenter());
        infowindow.open(map);
        }
    });
}
// # CTRL PRESSED #
var ctrlPressed = false,
    selectedMarkers = [];
window.onkeydown = function(e) {
  ctrlPressed = ((e.keyIdentifier == 'Control') || (e.ctrlKey ==
true));
}
window.onkeyup = function(e) {
  ctrlPressed = false;
}
```